REMARKS

Applicant's claims 10 and 15 objection because of typo and grammatical errors are corrected by the amendments to these claims wherein "compiling" replaces "computing" in claim 10 and second mentioned " step generated possible solutions" is removed in claim 15.

Claims 2, 13 and 14 are rejected under 35 USC 112, second paragraph, as being indefinite. Claim 2 is amended to remove the indefiniteness by the claim to read for "said two or more different sets of compiler options includes compile for best computation time, compile for best code size, and compile for intermediate points between best computation time and best code size." Claims 13 and 14 are amended to overcome the indefiniteness by calling for "said generating sets of useful solutions."

Claims 1,3, 6-11,14,15 and 22-23 are rejected under 35 U.S.C. 102 (b) as being anticipated by Granston et al. (U.S. Patent No. 5,966,538; hereinafter Granston).

The Grandson reference describes a process whereby the user builds (compiles) a program, executes it and collects detailed performance information about the execution of the program, and uses that performance information to determine what compiler options to use in subsequent builds. This process is iterative. In the Granston method, optionally with the user's help examines attributes of a previous execution, and suggests NEW options that should be used in subsequent builds. In other words, it is an expert system in which the compiler makes explicit recommendations, based on performance characteristics of your program, on exactly which options you should be compiling with. In effect, the compiler is looking for particular runtime characteristics which the compiler can exploit in subsequent builds, using the right option set. The implication is that the invention explicitly understands the relationship between a particular option and a particular performance characteristic of the target program. This is fundamentally different than applicant's invention.

In applicant's invention, the user is required to independently compile and run the program at least twice. Each build/run is completely different. In these remarks I refer to this run as a "trial" run. The options used in each trial must be different, but otherwise the exact meaning and semantics of the options specified by the user are irrelevant to this invention. Note that in each trial, all the functions in the program are compiled with the exact same option set.

For each trial, for each function in the program, we note the total time spent in the function, and the code size of the function, and finally the compiler options used during that trial. When we are done with all the "trials" we can build a table that contains, for each function in the program, for each trial, how much time was spent in it, and how large it was. For example we have something like the table below for three trials:

Trial	Function	build options	cycles	code size
1	main	-02	1234	90
2		-03	889	98
3		-03-ms	1090	100
1	func	-02	20	12
2		-03	22	12
3		-03-ms	18	10
1	func2	-02	178	213
2		-03	140	210
3		-03-ms	130	210

After we have collected all this information, the user in the present invention asks the user for explicit time constraints such as the program must run in 24024 cycles and/or the program must fit in 9400 bytes. Given the generated information, and given the user constraints, the invention can, using integer programming technique, figure out which trial option set to use to compile each function in the entire program such that the time or space constraint will be met. To illustrate, it might decide, for the above example, to compile "main" using trial 1 options, "func" using trial 3 options, and "func2" using trial 3 options, giving a cycle time of 1382 (1234+18+130), and a total code size of 310 bytes (90+10+210).

Applicants do not iterate through a user. Granston does. In applicant's teaching the user indicates what option sets are allowable and applicant's method and apparatus determines the most appropriate options for a function based on measured response to the option sets. Granston only suggests options. Applicant teaches to apply options at the function level. Granston does not. Applicant teaches optimizing an application characteristic given constraints on an alternate characteristic. Applicant's approach involves the user only to set the characteristic constraint or to choose a solution when the method has completed solving the optimization problem. Granston is analyzing the compiled code and is suggesting alternate or new options that may or may not be beneficial to the application, after which their method iterates with the user.

Applicant's amended claim 1 calls for "A method for automatically compiling an application program in a manner that optimizes one application characteristic, subject to limits on another application characteristic , comprising: compiling said application with two or more ~~a~~ different sets of compiler options that optimizes one application characteristic, subject to limits on another application characteristic to provide two or more executables, generating profile information from said executables, generating optimal solutions using said profile information, and automatically selecting and applying function level compiler options for said application program based upon said optimal solutions information."

This process is not taught or suggested in the Granston reference. It is not seen where Granston teaches compiling said application with a different set of compiler options that optimizes one application characteristic, subject to limits on another application characteristic to provide two or more executables and generating profile information from these executables. The Granston reference discusses utilizing information obtained from interviewing compiler user, compile time information obtained during one or more computations of the program and profile information collected at run time. There is no compiling said application with a different set of compiler options that optimizes one application characteristic, subject to limits on another application characteristic in Granston. There is no generating profile information from these executables in Granston. There are no generating optimal solutions using this profile information in Granston. Further there is no automatically selecting and applying function level compiler options for said application program based upon said optimal solutions information."

Claim 1, as amended, is therefore deemed allowable over Granston.

Claims 2- 6 and 24 dependent on claim 1 are deemed allowable for at least the same reasons as claim 1.

Claim 2 dependent on claim 1 is deemed allowable over Granston for at least the same reasons. Claim 2 amended calls for "said two or more different sets of compiler options includes compile for best computation time, compile for best code size, and compile for intermediate points between best computation time and best code size." Granston does not mention code size or compile for intermediate points between best computation time and best code size. The examiner has for claims 2, 4 and 5 applied Hejlsberg et al. U.S, Patent No. 5,535,391(hereinafter Hejlsberg). While Hejlsberg mentions compiling for the fastest code possible and the smallest code possible, Hejlsberg does not teach generating profile information from these executables and generating optimal solutions using this profile information and further, there is no automatically selecting and applying function level compiler options for said application program based upon said optimal solutions information.

Claim 3 amended calls for similar limitations as claim 1 and further calls for "compiling said application program with a first set of compiler options in a manner that optimizes one application characteristic, subject to limits on another application characteristic to provide a first executable, compiling said application program with a second set of computer options in a manner that optimizes said other application characteristic, subject to limits on said one application characteristic to provide a second executable." Claim 3 calls for generating and displaying sets optimal solutions from said profile information wherein the sets have methods of compiling at the function level in a solution space generator and automatically selecting and applying function level compiler options for said application program based upon selected optimal solutions so as to optimize said application program. This is not taught in Granston. Claim 3 is therefore deemed patentable over the Granston patent. While Hejlsberg mentions compiling for the fastest code possible and the smallest code possible, Hejlsberg does not teach generating profile information from these executables and generating optimal solutions using this profile information and further, there is no automatically selecting and applying function level compiler options for said application program based upon selected optimal solutions so as to optimize said application program.

Claim 24 is dependent on claim 1 and further calls for said application characteristic includes computation time, code size and power consumption.

Claims 4-6 dependent on claim 3 are deemed allowable for at least the same reasons as claim 3. Claim 4 amended calls for the option that is best for computation time and claim 5 amended for best code size. For claim 6 there is no teaching in the references of the step of analyzing the profile information against user supplied application characteristic constraints for selecting said compiler options by function.

Claim 10 amended calls for "compiling said application with a different set of compiler options that optimizes one application characteristic, subject to limits on another application characteristic to provide two or more executables; generating profile information from said executables; applying said profile information to a solver; generating sets of useful solutions from said solver wherein the sets have methods for compiling each function; and selecting a solution for said application program using said useful solutions for subsequent compiling of said application." This combination is not taught or suggested in Granston. It is not seen where Granston teaches compiling said application with a different set of compiler options that optimizes one application characteristic, subject to limits on another application characteristic to provide two or more executables and generating profile information from these executables. The reference discusses utilizing information obtained from interviewing compiler user, compile time information obtained during one or more computations of the program and profile information collected at run time. There is no solver in Granston that generates sets of useful solutions from said solver wherein the sets have methods for compiling each function. Granston does not mention sets of useful solutions at the function level. Further there is no selecting a solution for said application program using the useful solutions for subsequent compiling of said application.

Claims 11-19 and claim 25 dependent on claim 10 are deemed allowable for at least the same reasons as claim 10. Claim 11 amended further calls for "said selecting step includes displaying said useful solutions."

Claim 12 amended further calls for "said generating <u>sets of useful</u> <u>solutions</u> step includes generating an efficient frontier curve of optimum solution points and displaying said curve of solution points." Clearly, this is not taught or suggested in Granston. The examiner cites Fig. 5 of Bonadio (U.S. Patent No. 5,189,633; hereinafter Bonadio). At most Bonadio shows a graphic representation of an equation. It does not teach generating an efficient frontier curve of optimum solution points or displaying the curve of optimum solution points. There is nothing in the combination of the references to teach this.

Claim 13 amended further calls for "said generating sets of useful solutions step includes a zoom window of a section of said curve of solution points. Clearly this is not taught in Granston or Bonadio for the same reasons as claim 12.

Claim 14 amended further calls "said generating sets of useful solutions step includes linear programming and heuristics to reduce the number of permutations of option sets per function."

Claim 15 amended further calls for "said generating solutions step generates possible solutions and filters out the possible solutions that are not better in at least one application characteristic." This is not taught in Granston.

Claim 16 is dependent on claim 15 and is deemed allowable for at least the same reasons as claim 15. Claim 16 amended further calls for "said generating solutions step includes a search tree wherein each candidate is applied to a node and compared to the solution at the node and if faster in time and smaller in size replacing that candidate at the node, if neither faster nor smaller in size discarding the candidate, if faster only processing down the tree in one direction and if smaller only processing down the tree in a different direction." Clearly, this is not taught in Granston. The examiner cites Poutanen et al (U.S. Patent No. 5,978,795; hereinafter Poutanen). Poutanen discloses a search tree. Neither Granston nor Poutanen teach or suggest the combination that the examiner suggests. Neither teaches or suggest a search tree in connection with compiling an application or selecting compiling candidates or wherein the candidates are for code size and computation time or each candidate is applied to a node and compared to the solution at the node and if faster in time and smaller in size replacing that candidate at the node, if neither faster nor smaller in size discarding the candidate, if faster only processing down the tree in one direction and if smaller only processing down the tree in a different direction." Nothing like this is mentioned or suggested in either reference.

Claim 17 amended calls for "the step of selecting a solution includes displaying solution points on said solution point curve illustrating the application characteristic tradeoff that can be made by compiling each function as prescribed by said solution." Clearly, this is not taught in Granston for the reasons discussed above. The examiner cites

Fig. 5 of Bonadio (U.S. Patent No. 5,189,633; hereinafter Bonadio). At most Bonadio shows a graphic representation of an equation. It does not teach displaying solution points on a solution point curve illustrating the application characteristic tradeoff that can be made when compiling each function by the solution. There is nothing like this in either reference.

Claim 18 amended calls for "means for overriding the solvers choice of compiler options for a particular function.." Clearly, this is not taught in Granston for the reasons discussed above. The examiner cites Fig. 5 of Bonadio (U.S. Patent No. 5,189,633; hereinafter Bonadio). At most Bonadio shows a graphic representation of an equation. Neither Granston nor Bonadio teach the overriding means claimed.

Claim 19 dependent on claim 17 is deemed allowable for at least the same reasons as claim 17. Claim 19 amended calls for "compiling the application using chosen solution's set of function options.

Clearly, this is not taught in Granston for the reasons discussed above. The examiner cites Fig. 5 of Bonadio (U.S. Patent No. 5,189,633; hereinafter Bonadio). At most Bonadio shows a graphic representation of an equation.

Claim 25 dependent on claim 10 is deemed allowable for at least the same reasons as claim 10. Claim 25 further calls for "selecting a solution includes applying a user constraint to one application characteristic and automatically selecting the solution point that meets said constraint and optimizes another application constraint."

Claim 26 calls for "An apparatus for automatically generating an optimal code for an application where conflicting characteristics exist comprising: means for compiling said application with a compiler option to constrain a first characteristic and optimize the other characteristic and means for compiling said application with a compiler option to constrain said other characteristic and optimize the first characteristic to provide solutions and means for automatically selecting compiler options and compiling for said application program based upon said solutions." As discussed previously Granston does not teach such compiling criteria or selecting compiler options based solutions from on this type of compiling.

Claim 27 calls for "said first characteristic is size (or bytes) and the other characteristic is computation time (cycles)." This is not taught in the Granston patent.

Claim 28 calls for "said means for automatically selecting compiler options includes means for inputting an upper size limit for said application and said means for automatically selecting compiler options and compiling generates a solution that will minimize the computation time of the application subject to the size limit." This is not taught in the Granston patent.

Claim 29 calls for "includes means for inputting an upper computation time limit for said application and said means for selecting compiling compiler options and compiling generates a solution that will minimize the size of the application subject to the time limit." This is not taught in the Granston patent.

Claim 30 calls for "An apparatus for automatically generating an optimal code for an application where conflicting characteristics exist comprising: means for compiling said application with a compiler option to constrain a first characteristic and optimize the other characteristic; means for compiling said application with a compiler option to constrain said other characteristic and optimize the first characteristic to provide a set of minimal solutions for the application, and means for displaying and selecting said minimal solutions." As discussed previously the compiling steps are not taught in Granston.

Claim 31 calls for "The apparatus of claim 30 wherein each solution in the set of minimal solutions is such that the apparatus cannot compile an alternate solution that is both smaller in size and in computation time." This is not taught in the Granston patent. The combination of smaller in size and in computation time is not discussed in Granston.

Claim 32 calls for "A method for automatically generating an optimal code for an application comprising: compiling, measuring and recording size and computation time of the functions in an application while varying the compiler options to get size versus computation time behavior data for each function compiled with a set of compiler options, and means for generating a set of minimal solutions for the application such that the compiler cannot generate an alternate solution that is both smaller in size and in computation time." This is not taught in Granston. Granston does not teach compiling, recording and computation time of function in an application while varying the compiler options to get size versus computation time behavior data for each function compiled with a set of compiler options. Size versus computation time is not even mentioned in Granston. Further, Granston does not even mention generating a set of minimal solutions for the application such that the compiler cannot generate an alternate solution that is both smaller in size and in computation time.

Claim 33 dependent on claim 32 is deemed allowable for at least the same reasons as claim 32.

In view of the above applicant's claims 1-6, 10-19, and 24-33 are deemed allowable over the references.

The examiner states that claims 1-11, 17 and 20-23 are provisionally rejected for obvious type double patenting. Since the claims of application serial no. 09/510,217 are not yet patented there can be no double patenting. Further, since applicant's claims herein are not allowed there can be no double patenting. Further, applicant's allowed claims may be patently distinct from those in application serial no. 09/510,217. Applicants will therefore hold in abeyance the filing of a terminal disclaimer.

In view of the above claims 1-6, 10-19, and 24-33 are deemed allowable.

Respectfully submitted;
[illegible]
Robert L. Troike (Reg. 24183)

Telephone No.(301) 259-2089